RECEIVED
CENTRAL FAX CENTER

OCT 03 2005

KENYON & KENYON
Intellectual Property Law

RiverPark Towers
333 West San Carlos Street
Suite 600
San Jose, CA 95110
Direct Tel: (408) 975-7970
Facsimile: (408) 975-7501
sneal@kenyon.com

FACSIMILE TRANSMITTAL SHEET

TO: Examiner Tonia L. MEONSKE	FROM: Stephen T. Neal
COMPANY: USPTO	DATE: October 3, 2005
FAX NUMBER: (571) 273-8300	TOTAL NO. OF PAGES INCLUDING COVER: 21
PHONE NUMBER:	SENDER'S REFERENCE NUMBER: Intel 2207/6786
RE: Serial No.: 09/608,512	YOUR REFERENCE NUMBER: Group Art Unit: 2183

☐ URGENT ■ FOR REVIEW ☐ PLEASE COMMENT ☐ PLEASE REPLY ☐ CONFIRMATION

☐ ORIGINAL WILL FOLLOW ■ ORIGINAL WILL NOT FOLLOW

Notes/Comments: **APPEAL BRIEF**

1. Fax Cover Sheet (1)
2. Transmittal Form (1)
3. Fee Transmittal (and one copy) (2)
4. Appeal (17)

Total: (21) pages

Certificate of Facsimile Transmittal

I hereby certify that the above referenced correspondence is being transmitted via facsimile under 37 C.F.R. §1.11 to Examiner Tonia L. MEONSKE at facsimile number: (5713) 273-8300 at the United States Patent and Trademark Office.

Dated: October 3, 2005

Signature: Barbara Vance
Barbara Vance

75712_1.DOC

TRANSMITTAL FORM

(to be used for all correspondance after initial filing)

Total Number of Pages in this Submission	21

Application Number	09/608,512
Filing Date	June 30, 2000
First Named Inventor	Reynold V. D'SA et al.
Art Unit	2183
Examiner Name	Tonia L. MEONSKE
Attorney Docket Number	Intel 2207/6786

ENCLOSURES *(check all that apply)*

- ☒ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Reply
  - ☐ After Final
- ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) ____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☒ Appeal Communication to TC (Appeal Notice, Brief, Reply Brief)
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:
  1. **Facsimile Cover Sheet**

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm	Kenyon & Kenyon		
Signature	Stephen T. Neal		
Printed Name	Stephen T. Neal		
Date	October 3, 2005	Reg. No.	47,815

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. **M/S: APPEAL BRIEF – PATENTS – Via Facsimile**

Signature	Barbara Vance		
Typed or printed name	Barbara Vance	Date	October 3, 2005

Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2005

Effective 10/01/2004. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) **$500.00**

Complete if Known	
Application Number	09/608,512
Filing Date	June 30, 2000
First Named Inventor	Reynold V. D'SA et al.
Examiner Name	Tonia L. MEONSKE
Art Unit	2183
Attorney Docket No.	Intel 2207/6786

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☒ Deposit Account:

Deposit Account Number: **11-0600**

Deposit Account Name: **Kenyon & Kenyon**

The Director is authorized to: *(check all that apply)*
☒ Charge fee(s) indicated below ☒ Credit any overpayments
☒ Charge any additional fee(s) or any underpayment of fee(s)
☒ Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	
SUBTOTAL (1)					($) 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20 **	=	X	50.00	=	
Independent Claims		-3 **	=	X	200.00	=	
Multiple Dependent				X		=	

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	50	2202	25	Claims in excess of 20
1201	200	2201	100	Independent claims in excess of 3
1203	360	2203	180	Multiple dependent claim, if not paid
1204	200	2204	100	** Reissue independent claims over original patent
1205	50	2205	25	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	500.00
1403	1,000	2403	500	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	500	2452	250	Petition to revive – unavoidable	
1453	1,500	2453	750	Petition to revive – unintentional	
1501	1,400	2501	695	Utility issue fee (or reissue)	
1502	490	2502	245	Design issue fee	
1503	660	2503	330	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17 (q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	790	2809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
1801	790	2801	395	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) ____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 500.00

SUBMITTED BY				Complete *(if applicable)*	
Name (Print/Type)	Stephen T. Neal	*Registration No. (Attorney/Agent)*	47,815	*Telephone*	(408) 975-7500
Signature	Stephen T. Neal			*Date*	October 3, 2005

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing this form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

RECEIVED
CENTRAL FAX CENTER
OCT 0 3 2005

Patent

Attorney Docket No.: Intel 2207/6786
Assignee: Intel Corporation

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANTS	:	Reynold v. D'SA et al.
SERIAL NO.	:	09/608,512
FILED	:	June 30, 2000
FOR	:	METHOD AND APPARATUS FOR THE DETECTION, RECOVERY, AND PREVENTION OF BOGUS BRANCHES
GROUP ART UNIT	:	2183
EXAMINER	:	Tonia L. MEONSKE

VIA FACSIMILE

M/S: APPEAL BRIEF – PATENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being facsimile transmitted to the United States Patent and Trademark Office via facsimile number: (571) 273-8300 or deposited with the United States Postal Service as first class mail in an envelope addressed to: M/S: APPEAL BRIEF - PATENT; Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on

Dated: October 3, 2005 Signature: Barbara Vance
Barbara Vance

ATTENTION: Board of Patent Appeals and Interferences

APPEAL BRIEF

Dear Sir:

This brief is in furtherance of the Notice of Appeal, filed in this case on August 3, 2005.

10/05/2005 MBINAS 00000016 110600 09608512
01 FC:1402 500.00 DA

1. REAL PARTY IN INTEREST

The real party in interest in this matter is INTEL CORPORATION of Santa Clara, California (Recorded September 7, 2000, Reel/Frame 011175/0931).

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals.

3. STATUS OF THE CLAIMS

Claims 1-26 are pending in the application. Claims 1-26 were rejected under 35 U.S.C. §103(a).

4. STATUS OF AMENDMENTS

Claims 1, 7, 10, 14, and 19 were amended in the previous response. The amendments were not entered by the Examiner.

5. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claim 1 recites a method of detecting, recovering from and preventing bogus branch instructions in a microprocessor. A first macro instruction is decoded into at least one micro-op. (*See* **Figure 1** and p. 4, lines 20-28, p. 6, lines 2-7). The at least one micro-op is written into a decoded micro-op cache 17. (*See* **Figures 1 and 2** and p. 4, lines 1-5, p. 6, lines 8-26). Branch prediction logic is used to predict whether the at least one micro-op is a branch. (*See* **Figure 2** and p. 6, lines 28-37). The at least one micro-op is executed. (*See* **Figure 2** and

p. 7, lines 11-14). It is determined if the at least one executed micro-op is a bogus branch of the first macro instruction and processing with a second macro instruction is continued. (*See* **Figure** 2 and p. 7, lines 27-30). If the at least one executed micro-op is determined to be a bogus branch, then any other micro-ops which pertain to the at least one executed bogus branch micro-op are flagged. (See p.4, lines 14-15). These flagged micro-ops are removed for retirement. (*See* **Figure 2** and p. 7, lines 25-26). Finally, a branch prediction logic storage buffer upon which the branch prediction logic is based is scrubbed. (*See* **Figure 2** and p. 8, lines 13-20).

Independent claim 7 recites a method of detecting bogus branch instructions in a microprocessor instruction pipeline. It is predicted whether a first micro-op is a bogus branch instruction. (*See* **Figure 2** and p. 6, lines 28-37). At least one second micro-op related to the first micro-op is found by looking ahead at the instruction pipeline. (*See* **Figure 2** and p. 6, line 37, p. 7, lines 1-5). If the first micro-op is predicted to be a bogus branch, a signal flag, indicating a bogus branch, is attached to the at least one second micro-op. (*See* **Figure 2** and p. 6, line 37, p. 7, lines 1-10).

Independent claim 10 recites a method of recovering from a bogus branch instruction in a microprocessor instruction pipeline. It is determined whether a first micro-op is a bogus branch. (*See* **Figure 2** and p. 7, lines 27-30). At least one second micro-op related to the first micro-op is deallocated from a decoded micro-op cache. (*See* **Figure 2** and p. 7, lines 31-37, p. 8, lines 1-12).

Independent claim 14 recites a method of preventing a bogus branch instruction from being executed in a microprocessor instruction pipeline. At least one micro-op is written into a decoded micro-op cache. (*See* **Figures 1 and 2** and p. 4, lines 1-5, p. 6, lines 8-26). The at least

one micro-op is retired. (*See* **Figure 2** and p. 7, lines 25-26). Entries from a branch prediction logic storage buffer that would later produce bogus branches are removed. (*See* **Figure 2** and p. 8, lines 18-20).

Independent claim 19 recites an apparatus for detecting, recovering from and preventing bogus branch instructions in a microprocessor, and contains the following elements. A decoded micro-op cache 16 into which are written at least one decoded micro-op of a macro instruction. (*See* **Figure 3** and p. 9, lines 16-26). A branch prediction logic storage buffer 21 for predicting whether a branch will be taken upon execution of the at least one decoded micro-op. (*See* **Figure 3** and p. 9, lines 26-28). An instruction execution unit 41 for executing the at least one micro-op. (*See* **Figure 3** and p. 9, lines 36-37, p. 10, lines 1-3). An instruction retirement unit 51 which determines whether the at least one micro-op is of a bogus branch macro instruction. If the instruction retirement unit 51 determines the at least one micro-op is of a bogus branch macro instruction, any other micro-ops stored in the decoded micro-op cache 16 pertaining to that bogus branch macro instruction are flagged and removed to the instruction retirement unit for retirement and the branch prediction logic storage buffer is scrubbed. (*See* **Figure 3** and p. 10, lines 6-13).

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 7-13 were rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle et al., U.S. Patent Number 5,956,495 (hereinafter "Kahle") in view of McCrocklin et al., U.S. Patent Number, 4,761,733 (hereinafter "McCrocklin").

B. Claims 1-6 and 14-26 were rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of McCrocklin in view of Shiell, U.S. Patent Number 5,864,697 (hereinafter "Shiell").

7. ARGUMENT

A. Claims 7-13 were rejected under 35 U.S.C. §103(a) as being unpatentable over Kahle in view of McCrocklin.

Independent claims 7 and 10 recite a method of detecting bogus branch instructions in a microprocessor instruction pipeline. It is predicted whether a first micro-op is a bogus branch instruction. At least one second micro-op related to the first micro-op is found by looking ahead at the instruction pipeline. If the first micro-op is predicted to be a bogus branch, a signal flag, indicating a bogus branch, is attached to the at least one second micro-op. Claims 8 and 9 depend from claim 7, and claims 11-13 depend from claim 10.

Appellants respectfully submit that neither Kahle, McCrocklin, nor any combination thereof disclose predicting or determining whether a first micro-op is a bogus branch instruction, as recited in claims 7 and 10.

Merriam-Webster's Online Dictionary (www.m-w.com) defines "bogus" as being "not genuine, counterfeit, sham." In line with this definition and as the specification explains, a "bogus branch" is one that "...occur[s] at an address that does not contain a branch or ... [has] a target address that is invalid." (*See* p. 2, lines 8-12). Examiner's incorrect definition of "bogus branch" deviates wildly from Appellants' and takes two different, but similar forms throughout the Office Actions. In Office Action dated November 5, 2004, Examiner states:

> Examiner has interpreted "bogus branch" to mean a mispredicted branch instruction, or where a branch instruction is predicted to be taken or not taken, and the prediction is false.

(Office Action dated November 5, 2004, p. 15, paragraph 33).

In the next Office Action, Examiner reduces this definition to a "...branch instruction that is not taken..." and gets to this interpretation through the following flawed logic:

> A bogus branch instruction in claims 7-13 is interpreted to be a branch instruction that is not taken. Merriam-Webster's online dictionary defines "bogus" as "not genuine" and "genuine" as "actual or true." Since "Genuine" is interpreted to be "actual", "not genuine" in interpreted to be "not actual." Therefore something that is "bogus" is interpreted to be something that is "not actual." A branch instruction that is not taken is not an actual branch instruction to a non-sequential address, i.e. the branch instruction is not actually branching. Therefore a not taken branch instruction is a bogus, or not an actual, branch instruction. So Kahle has in fact taught *predicting whether a first micro-op is a bogus branch instruction* (Kahle predicts whether a first micro-op is taken or not-taken. Column 9, lines 56-65 and column 11, lines 1-35). Therefore this argument is moot.

(Office Action dated May 3, 2005, pp. 2-3, paragraph 5).

As conceded by Examiner, Kahle "predicts whether a first micro-op [branch] is taken or not taken." (*See* Office Action dated May 3, 2005, p. 3, paragraph 5). To apply Examiner's misguided definition of "bogus branch" to the situation described in Kahle is contrary to the plain meaning of the word "bogus" when applied directly to the term "branch instruction" and belies the fact that a branch instruction in Kahle is still present even if that branch is predicted to be not taken, despite Examiner's conclusion of the opposite. Surely the branch instruction does not instantly morph into some other instruction simply because it is predicted not to branch, just as an ADD instruction does not become a "non-ADD" instruction when a subprogram that contains it is not reached because a branch was not taken. Furthermore, Examiner's conflated logic, as applied to Kahle, describes a *prediction* that is bogus and not the branch itself.

Appellants respectfully submit, therefore, that elements of claims 7 and 10 are neither shown nor suggested by the cited references. Claims 8-9 and 11-13 depend from claims 7 and 10. Accordingly reconsideration and withdrawal of the rejection of claims 7-13 under 35 U.S.C. §103(a) is respectfully requested.

B. Claims 1-6 and 14-26 were rejected under 35 U.S.C. §103(a) as being unpatentable over Kahle in view of McCrocklin in view of Shiell.

Independent claims 1, 14, and 19 recite a method of, and apparatus for, detecting, recovering from and preventing bogus branch instructions in a microprocessor. A first macro instruction is decoded into at least one micro-op. The at least one micro-op is written into a decoded micro-op cache. Branch prediction logic is used to predict whether the at least one micro-op is a bogus branch. The at least one micro-op is executed. It is determined if the at least one executed micro-op is a bogus branch of the first macro instruction and processing with a second macro instruction is continued. If the at least one executed micro-op is determined to be a bogus branch, then any other micro-ops which pertain to the at least one executed bogus branch micro-op are flagged. These flagged micro-ops are removed for retirement. Finally, a branch prediction logic storage buffer upon which the branch prediction logic is based is scrubbed. Claims 2-6 depend from claim 1, claims 15-18 depend from claim 14, and claims 20-26 depend from claim 19.

Appellants respectfully submit that neither Kahle, McCrocklin, Shiell, nor any combination thereof disclose determining whether a first micro-op is a bogus branch instruction, as recited in claims 1 and 19. As shown above, Kahle, McCrocklin do not disclose this element. Further, Shiell does not disclose this element.

Appellants respectfully submit that neither Kahle, McCrocklin, Shiell, nor any combination thereof disclose removing entries from a branch prediction logic storage buffer that would later produce bogus branches, as recited in claim 14. As shown above, Kahle, McCrocklin and Shiell do not disclose bogus branches in any way.

Appellants respectfully submit, therefore, that elements of claims 1, 14, and 19 are neither shown nor suggested by the cited references. Claims 2-6, 15-18, and 20-26 depend from claims 1, 14, and 19. Accordingly reconsideration and withdrawal of the rejection of claims 1-6 and 14-26 under 35 U.S.C. §103(a) is respectfully requested.

The Examiner is hereby authorized to charge any additional fees, which may be necessary for consideration of this paper to Kenyon & Kenyon Deposit Account No. **11-0600.**

Respectfully submitted,

KENYON & KENYON

Date: October 3, 2005

By: Stephen T. Neal
Stephen T. Neal
(Reg. No. 47,815)

KENYON & KENYON
333 West San Carlos St., Suite 600
San Jose, CA 95110

Telephone: (408) 975-7500
Facsimile: (408) 975-7501

APPENDIX

(Brief of Appellant Reynold v. D'SA et al.
U.S. Patent Application Serial No. 09/608,512)

8. CLAIMS ON APPEAL

1. A method of detecting, recovering from and preventing bogus branch instructions in a microprocessor, the method comprising:

decoding a first macro instruction into at least one micro-op;

writing the at least one micro-op into a decoded micro-op cache;

predicting by branch prediction logic whether the at least one micro-op is a branch;

executing the at least on micro-op;

determining if the at least one executed micro-op is a bogus branch of the first macro instruction; and

continuing processing with a second macro instruction,

wherein if the at least one executed micro-op is determined to be a bogus branch, then the method further comprises:

flagging any other micro-ops which pertain to the at least one executed bogus branch micro-op;

removing the flagged micro-ops for retirement; and

scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based.

2. The method according to claim 1, further comprising:

fetching from a main memory the macro instruction.

3. The method according to claim 1, wherein the at least one micro-op is written into the decoded micro-op cache in an order a branch table buffer predicts that the at least one micro-op should be executed.

4. The method according to claim 1, wherein executing the at least one micro-op is in at least one of an in-order or out-of-order fashion.

5. The method according to claim 1, wherein scrubbing the branch prediction logic storage buffer further comprises at least one of:

deallocating any other micro-ops pertaining to the at least one executed bogus branch micro-op;

deallocating at least one old set which had been overwritten in the decoded micro-op cache by a built instruction trace;

deallocating at least one entry that is related to a branch in at least one old set in the decoded micro-op cache; and

deallocating at least one entry that is related to a branch of at least one old set in the decoded micro-op cache that is downstream from the at least one executed bogus branch micro-op.

6. The method according to claim 1, further comprising:

determining if the branch has been taken.

7. A method of detecting bogus branch instructions in a microprocessor instruction pipeline, the method comprising:

predicting whether a first micro-op is a bogus branch instruction; and

looking ahead in the instruction pipeline to at least one second micro-op related to the first micro-op,

wherein if the first micro-op is predicted to be a bogus branch, the method further comprises;

attaching a signal flag that indicates a bogus branch to the at least one second micro-op.

8. The method according to claim 7, further comprising:

decoding at least one macro instruction into the first micro-op and the at least one second micro-op; and

writing the first micro-op and the at least one second micro-op into a decoded micro-op cache.

9. The method according to claim 7, wherein the prediction of whether the first micro-op is a bogus branch instruction is based on branch prediction logic.

10. A method of recovering from a bogus branch instruction in a microprocessor instruction pipeline, the method comprising:

determining whether a first micro-op is a bogus branch; and

deallocating from a decoded micro-op cache at least one second micro-op related to the first micro-op.

11. The method according to claim 10, wherein determining whether the first micro-op is a bogus branch is based on branch prediction logic.

12. The method according to claim 10, wherein deallocating from the decoded micro-op cache the at least one second micro-op is accomplished by checking whether a bogus branch signal flag has been attached to the at least one second micro-op.

13. The method according to claim 10, wherein deallocating further comprises at least one of:

removing the specific bogus branch;

removing all branches in a set with the bogus branch;

removing all branches in the decoded micro-op cache; and

clearing the entire decoded micro-op cache.

14. A method of preventing a bogus branch instruction from being executed in a microprocessor instruction pipeline, the method comprising:

writing at least one micro-op into a decoded micro-op cache;

retiring the at least one micro-op; and

removing entries from a branch prediction logic storage buffer that would later produce bogus branches.

15. The method according to claim 14, wherein retiring the at least one micro-op comprises at least:

determining what the actual result for the retired at least one micro-op was.

16. The method according to claim 14, wherein scrubbing the branch prediction logic storage buffer comprises at least:

comparing what an actual result of the retired at least one micro-op is to an instruction trace in the branch prediction logic storage buffer.

17. The method according to claim 14, wherein removing entries from the branch prediction logic storage buffer further comprises at least one of:

deallocating any other micro-ops pertaining to the at least one retired micro-op;

deallocating at least one old set which had been overwritten in the decoded micro-op cache by a built instruction trace;

deallocating at least one entry that is related to a branch in at least one old set in the decoded micro-op cache; and

deallocating at least one entry that is related to a branch of at least one old set in the decoded micro-op cache that is downstream from the at least one retired micro-op.

18. The method according to claim 14, wherein removing entries can be accomplished at the time of at least one of writing or retiring.

19. An apparatus for detecting, recovering from and preventing bogus branch instructions in a microprocessor, the method comprising:

a decoded micro-op cache into which are written at least one decode micro-op of a macro instruction;

a branch prediction logic storage buffer for predicting whether a branch will be taken upon execution of the at least one decoded micro-op;

an instruction execution unit for executing the at least one micro-op; and

an instruction retirement unit which determines whether the at least one micro-op is of a bogus branch macro instruction,

wherein if the instruction retirement unit determines the at least one micro-op is of a bogus branch macro instruction,

any other micro-ops stored in the decoded micro-op cache pertaining to that bogus branch macro instruction are flagged and removed to the instruction retirement unit for retirement

and the branch prediction logic storage buffer is scrubbed.

20. The apparatus according to claim 19, further comprising:

a main memory in which the macro instruction is stored; and

an instruction fetch unit for fetching the macro instruction from the main memory.

21. The apparatus according to claim 19, further comprising:

an instruction decode unit for translating the macro instruction into the at least one decoded micro-op.

22. The apparatus according to claim 19, further comprising:

a jump execution unit which determines whether a branch was taken upon execution of the at least one decoded micro-op.

23. The apparatus according to the claim 19, wherein the branch prediction logic storage buffer applies branch prediction logic to predict whether a branch will be taken upon execution of the at least one decoded micro-op.

24. The apparatus according to claim 19, wherein if the branch prediction logic storage buffer predicts a branch will be taken upon execution of the at least one decoded micro-op, an instruction trace is built pertaining to the predicted branch.\

25. The apparatus according to claim 24, wherein the built instruction trace is inserted into the decoded micro-op cache such that the micro-ops of the branch macro-instruction are executed.

26. The apparatus to claim 19, wherein the branch prediction logic storage buffer is scrubbed by deallocation of at least one of

any other micro-ops pertaining to the bogus branch macro instruction,

any old set which had been overwritten in the decoded micro-op cache by a built instruction trace,

all entries that are related to any branches in the old set, and

all entries that are related to the branches in the old set that are downstream from the retired branch macro instruction.

9. EVIDENCE APPENDIX

No further evidence has been submitted with this Appeal Brief.

10. RELATED PROCEEDINGS APPENDIX

Per Section 2 above, there are no related proceedings to the present Appeal.